(1). Process Based Multitasking V/s Thread Based Multitasking

Process Based Multi-tasking	Thread Based Multi-tasking
➢ A process is essence of program that is executing which running parallel	➢ Thread is a such part of multithreaded program which defines separate path for execution
➢ It allows you to run java compiler and text editor at a same time.	➢ It doesn't support java compiler and text editor run simultaneously. Both have performed different tasks.
➢ Process multitasking has more overhead than thread multitasking.	➢ Thread multitasking has less overhead than process multitasking.
➢ Here, it is unable to gain access over idle time of CPU.	➢ It allows taking gain access over idle time taken by CPU.
➢ It is not under control of java.	➢ It is totally under control of java.
➢ Process based multitasking is comparatively heavyweight process comparing to thread based multitasking.	➢ Thread based multitasking is known to be lighter eight process.
➢ Inter-process communication is expensive and limited.	➢ Inter-thread communication is inexpensive and context switching from one thread to other.
➢ It has slower data rate multitasking.	It has faster data rate multithreading or tasting.

(2). Method Overloading V/s Method Overriding

Method overloading	Method overriding
➢ It occurs when two or more method have same name with different signature.	➢ Overriding occurs when a class declares a method that has same type of signature.
➢ It is used in java program frequently because it allows using the same name for group of method that basically has same purpose.	➢ Overriding is a compatibility of java program because each overridden method has provided for unique implementation.
➢ Overloading allows programmer to use different functionalities to a same named method.	➢ Overriding allows programmer to add other functionalities to its method is classes.
➢ Java complier issues on error message because it is unable to determine which form to use.	➢ Java compilers issues an error message it method overrides another method has different return type.
➢ It allows easy handle to default parameters.	➢ It uses super class reference to “super class” object.
➢ It can occur within a class in different method.	➢ Overriding requires super & sub class for different method.
➢ Println ( ) is example of overloading.	➢ Versions of Java are an example of overriding.
➢ Constructor can be overloaded.	➢ Constructor can’t be overridden.

(3). Byte Stream V/s Character Stream

Byte stream	Character stream
➢ Byte stream provides convenient means for handling I/O if byte.	➢ Character stream provides convenient means for handling I/O character.
➢ Byte stream were included in java 1.0 version.	➢ Character stream were included in 1.1 versions.
➢ At the top there are two main abstract classes input stream & output stream.	➢ At the top there are two main abstract classes’ reader & writer.
➢ Concrete subclasses to handle devices like file, n/w	➢ It has to handle Unicode character streams.

connection and memory buffers.	
➢ Read ( ) & write ( ) methods respectively read & write byte of data. These methods are overridden from byte stream class.	➢ Read ( ) & write ( ) methods respectively use read and write character at data. These methods are overridden from character stream class.
➢ Both provide API & partial implementation for input stream output stream whose size is 8 but byte.	➢ Both provide API & partial implementation for reader. Writer whose size is 16 but character.
➢ Handle text file.	➢ Handle binary file.

(4). Type Casting V/s Type Conversion

Type casting	Type conversion
➢ Type casting takes place between incompatible types.	➢ Type conversion can take place in both compatible & incompatible types.
➢ In java there is no such concept of automatic type casting.	➢ In java type conversion is automatically done between compatible types.
➢ Type casting is allowed explicitly.	➢ Type conversion is allowed implicitly.
➢ Wrapper classes and special methods are used in type casting.	➢ Narrowing & widening conversion method are used in type conversion.

(8). Vector V/s Array

Vector	Array
➢ Convenient to use & store object.	➢ Convenient to use simple data types.
➢ Size can be varied for strongly object.	➢ Size can be explicitly specified.
➢ Different types of objects are stored.	➢ Save data type’s based values are stored.
➢ Vector has no dimension.	➢ Arrays do have fix dimension.
➢ By converting simple type to	➢ Arrays cannot support this

object by using wrapper class.	particular feature.

(9). String V/s String Buffer

String	String Buffer
➢ String represents fixed length, immutable character sequences.	➢ String buffer represents grow-able & writable character sequence.
➢ String argument stores characters without reallocation.	➢ String buffer allocates reserves room for 16 characters. It uses extra fragmentation for
➢ String can be use as primitive data type.	➢ String buffer are not used as data type.
➢ Literals directly can be assigning to string.	➢ String buffer can't use literals as string directly.
➢ String doesn't have constructors to define.	➢ String buffer defines following • string buffer ( ) • string buffer (int size) • string buffer (string s)

(10). Interface V/s Abstract Class

Interface	Abstract class
➢ Interface is use to be implemented in java programs.	➢ Abstract class is use to extend in java program.
➢ Object can be taken of interface.	➢ Object can not be considered.
➢ Interface is use as reusability of codes.	➢ Abstract class is not useful as reusability as codes.
➢ Can be define & declare in base class.	➢ Method can be declare in abstract class and must be used in subclass.
➢ Syntax of Interface: - ***interface <name>*** ***{*** ***variables method.*** ***}***	➢ Syntax of Abstract Class: - ***abstract class <class name>*** ***{*** ***variable method.*** ***}***